Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

Scheduling for multifunction radar via two-slope benefit functions

Peter W. Moo

Defence R&D Canada – Ottawa

Technical Memorandum
DRDC Ottawa TM 2010-259
December 2010

Canada

Scheduling for multifunction radar via two-slope benefit functions

Peter W. Moo
Defence R&D Canada – Ottawa

Defence R&D Canada – Ottawa
Technical Memorandum
DRDC Ottawa TM 2010-259
December 2010

Principal Author

Original signed by Peter W. Moo

Peter W. Moo

Approved by

Original signed by D. Dyck

D. Dyck
Head/RS

Approved for release by

Original signed by C. McMillan

C. McMillan
Head/Document Review Panel

Abstract

The scheduling of tracking and surveillance looks for multifunction radar is considered. A sequential technique is proposed, whereby tracking looks are scheduled first, and surveillance looks are then scheduled to occupy gaps in the radar time line. The Two-Slope Benefit Function (TSBF) Scheduler is used to schedule the tracking looks and requires that each tracking look request has a benefit function, which specifies benefit as a function of start time. This method accounts for both look priority and target dynamics in formulating a look schedule. If the radar is overloaded with tracking look requests, the TSBF Scheduler down-selects a set of looks which can be scheduled, using a method which favours higher priority looks. Looks are scheduled to maximize the total benefit, and the resulting maximization is shown to be a linear program which can be solved efficiently using the Simplex Method. Given a tracking look schedule, the Gap-Filling Scheduler is used to schedule surveillance looks. A method for prioritising surveillance looks is proposed.

Résumé

On examine la planification des visées de suivi et de surveillance pour un radar multifonction. Une technique séquentielle est proposée, selon laquelle on planifie d'abord les visées de suivi, puis les visées de surveillance pour combler les trous dans l'échéancier du radar. Le planificateur à fonction d'avantage à deux pentes (Two-Slope Benefit Function, TSBF) sert à planifier les visées de suivi, et cela exige que chaque demande de visée de suivi ait une fonction d'avantage, qui spécifie l'avantage en fonction de la date/de l'heure de début. Cette méthode tient compte de la priorité des visées et de la dynamique des cibles pour formuler un échéancier des visées. Si le radar est surchargé de demandes de visées de suivi, le planificateur à TSBF réduit la priorité d'un ensemble de visées qui peuvent être planifiées, à l'aide d'une méthode qui favorise les visées prioritaires. Les visées sont planifiées de façon à maximiser l'avantage total et on démontre que la maximisation qui en résulte est un programme linéaire qui peut être résolu efficacement à l'aide de la méthode du simplex. Une fois qu'un échéancier de visées de suivi est établi, on utilise un planificateur complémentaire (Gap-Filling Scheduler, GFS) pour planifier les visées de surveillance. On propose une méthode de priorisation des visées de surveillance.

This page intentionally left blank.

Executive summary

Scheduling for multifunction radar via two-slope benefit functions

Peter W. Moo; DRDC Ottawa TM 2010-259; Defence R&D Canada – Ottawa; December 2010.

Modern naval radars use phased array antennas, which allow the radar beam to be controlled and adapted almost instantaneously. The flexibility of phased array technology allows the radar to carry out multiple functions, each of which place different demands on the radar. Each function, such as surveillance, target tracking and fire control, requests that a number of looks be carried out by the radar. The role of a radar resource manager is to receive the look requests from the various functions and formulate a schedule to be executed by the radar. An important aspect of this role is deciding which looks should be dropped in overload situations, when the radar does not have sufficient time line to execute all look requests. The scheduling of tracking and surveillance looks is considered.

This work proposes a scheduling method called the Sequential Scheduler, which consists of the Two-Slope Benefit Function (TSBF) Scheduler for tracking looks and the Gap-Filling Scheduler (GFS) for surveillance looks. Tracking looks are scheduled first, and surveillance looks are then scheduled to occupy gaps in the radar time line. Associated with each tracking look request is a benefit function, which quantifies benefit as a function of start time. This method accounts for both look priority and target dynamics in formulating a look schedule. If required, the TSBF Scheduler down-selects a set of look requests which can be scheduled, using a method which favours high priority looks. Looks are then scheduled to maximize the total benefit. For the case of a two-slope benefit function, it is shown that the maximization problem is a linear program that can be solved by the Simplex Method. Given a tracking look schedule, the GFS is used to schedule surveillance looks. A method for prioritising surveillance looks is proposed.

The proposed technique schedules look requests with arbitrary priorities and look parameters. Higher priority look requests are favoured during the computation of the schedule. For tracking look requests, larger values of peak benefit enhance the likelihood that a look will be down-selected. Larger values of slopes for early and late scheduling enhance the likelihood that a look will be scheduled closer to its desired start time. The proposed method for prioritising surveillance looks allows particular surveillance regions to be revisited more often than others. The use of the Simplex Method for scheduling enables the scheduler to process large numbers of look requests in a computationally efficient manner.

Sommaire

Scheduling for multifunction radar via two-slope benefit functions

Peter W. Moo ; DRDC Ottawa TM 2010-259 ; R & D pour la défense Canada – Ottawa ; décembre 2010.

Les radars navals modernes utilisent des antennes réseau à commande de phase, qui permettent de contrôler et d'adapter le faisceau radar presque instantanément. Grâce à la souplesse de la technologie à commande de phase, le radar peut exécuter de multiples fonctions qui imposent toutes des exigences différentes au radar. Chaque fonction, comme la surveillance, le suivi des cibles et la conduite de tir, exige qu'un certain nombre de visées soient effectuées par le radar. Le rôle d'un gestionnaire des ressources d'un radar est de recevoir les demandes de visées des différentes fonctions et de formuler un échéancier qui doit être suivi par le radar. Un aspect important de ce rôle est de choisir les visées qui doivent être laissées de côté lorsque le système est surchargé, lorsque le radar n'a pas le temps d'exécuter toutes les demandes de visées. On examine la planification des visées de suivi et de surveillance.

Ces travaux proposent une méthode de planification appelée planificateur séquentiel (Sequential Scheduler), composée du planificateur à fonction d'avantage à deux pentes (Two-Slope Benefit Function, TSBF) pour les visées de suivi et du planificateur complémentaire (Gap-Filling Scheduler, GFS) pour les visées de surveillance. On planifie d'abord les visées de suivi, puis on planifie les visées de surveillance pour combler les trous dans l'échéancier du radar. Une fonction d'avantage est associée à chaque demande de visée de suivi : elle quantifie l'avantage en fonction de la date/de l'heure de début. Cette méthode tient compte de la priorité des visées et de la dynamique des cibles pour formuler un échéancier des visées. Au besoin, le planificateur TSBF réduit la priorité d'un ensemble de demandes de visées qui peuvent être planifiées, à l'aide d'une méthode qui favorise les visées prioritaires. Les visées sont ensuite planifiées de façon à maximiser l'avantage total. Dans le cas d'une fonction d'avantage à deux pentes, on démontre que le problème de maximisation est un programme linéaire qui peut être résolu à l'aide de la méthode du simplex. Une fois qu'un échéancier de visées de suivi est établi, on utilise un planificateur GFS pour planifier les visées de surveillance. On propose une méthode de priorisation des visées de surveillance.

La technique proposée planifie les demandes de visées avec des priorités arbitraires et des paramètres de visées. Le système favorise les demandes de visées prioritaires pendant l'établissement de l'échéancier. Pour les demandes de visées de suivi, il est plus probable que la priorité d'une demande sera réduite si cette dernière a une valeur d'avantage maximal supérieure. De plus, il est plus probable qu'une visée sera planifiée à un moment proche de sa date/son heure de début souhaitées si elle a une grande valeur de pentes pour une

planification anticipée et tardive. La méthode proposée pour établir l'ordre de priorité des visées de surveillance permet de réobserver des régions de surveillance particulières plus souvent que d'autres. Grâce à l'utilisation de la méthode du simplex pour la planification, le planificateur peut traiter de nombreuses demandes de visées de façon efficiente.

This page intentionally left blank.

Table of contents

List of figures

List of tables

Acknowledgments

The author thanks Jack Ding for numerous helpful discussions on tracking and radar scheduling.

This page intentionally left blank.

1 Introduction

The use of phased array antennas has enhanced the flexibility and utility of radar. In particular, phased array technology allows the radar beam to be controlled and adapted almost instantaneously. This flexibility enables the radar to carry out multiple functions simultaneously, such as fire control, tracking and surveillance, where each function carries out a number of looks. The execution of multiple functions necessitates the study of radar resource management, which considers the prioritisation and scheduling of radar looks. Radar resource management is especially important in overload situations, when the radar does not have sufficient time line to schedule all requested looks. In this case, the radar scheduler must decide which looks should be scheduled and which should be dropped. Additionally, for the looks to be scheduled, a start time for each look must be determined.

Previous work on scheduling includes papers which consider single interval looks and those which consider coupled looks. A single interval look occupies one continuous interval on the radar time line, whereas a coupled look consists of a transmission interval, an idle interval and a reception interval. For single interval looks, Stafford [1] proposed a scheduling process for looks which have differing priorities. Vine [2] developed a scheduling algorithm where look priorities are computed using a fuzzy logic approach. In [3], looks were scheduled according to a time balance which is associated with each look. Sabatini and Tarantino [4] suggested strategies for scheduling prioritized targets when the radar is overloaded. In [5], two scheduling methods were proposed. The first is based on dynamic programming, which results in an optimal but computationally expensive solution. The second method is suboptimal but is computationally less demanding.

For the scheduling of coupled looks, looks may be interleaved to take advantage of the idle radar time between transmission and reception. Izquierdo-Fuente and Casar-Corredera [6] developed an optimal interleaving algorithm based on neural networks. Orman et al. [7] presented a heuristic-based approach to the scheduling of coupled looks. In [8], the prioritized scheduling of interleaved and non-interleaved looks was considered. Miranda et al. [9] carried out a comparison between the schedulers proposed by Butler [3] and Orman et al. [7].

Previous work on radar scheduling has considered the relative priorities of looks when formulating a radar schedule. Looks with higher priorities are more likely to be retained in overload situations. When not all looks can be scheduled at their desired start times, looks with higher priority are generally scheduled closer to their desired start times.

This paper proposes a technique called the Sequential Scheduler, which considers both look priorities and target dynamics in formulating a radar schedule. In overload situations, the Sequential Scheduler considers look priorities in deciding which looks to retain. For track updates, the error covariance varies with update time and target dynamics. The Sequential Scheduler accounts for individual target dynamics via the change in error covariance to

schedule track updates.

The paper is organised as follows. Section 2 formulates the scheduling problem to be considered and presents an overview of the Sequential Scheduler. In Section 3, the Two-slope Benefit Function (TSBF) Scheduler is described. Section 4 presents the Gap-Filling Scheduler (GFS). A number of scheduling examples are presented in Section 5. The examples are chosen to illustrate the properties of the proposed scheduling technique. Finally, conclusions are presented in Section 6.

2 Scheduling paradigm

To accurately state the goal of a radar scheduler, it is necessary to first present definitions of a function, a task and a look. The radar carries out multiple functions, which include weapon control, target tracking, and surveillance. Each function consists of one or more tasks. For the weapon control function, a task involves the control of an individual weapon. Similarly, for the target tracking function, a task involves the tracking of an individual target. The surveillance function monitors a specified region of interest. A surveillance task may include the monitoring of a subregion within the specified region of interest. The surveillance function can also be thought of as consisting of a single task, where the task involves monitoring the entire region of interest.

Each task consists of several looks, where a look requires one continuous time interval of finite duration to be completed. For a tracking task, a look is an attempt to update a track by steering the radar in the direction of the expected location of the target. In this case, a look could consist of one or more beam positions of the radar. For a surveillance task, a look could consist of a single beam position or multiple beam positions. Since a look has been defined to require a continuous time interval to be completed, it is beneficial to define surveillance looks to be as short in duration as possible. This allows the scheduler the flexibility to interleave looks from multiple tasks.

Each task sends look requests to the radar scheduler. For a target tracking task, a look request may consist of an attempt to update a track at a specified time. The specified time will depend on the time of the track update, the estimated target dynamics and the tracking model. For all tasks, look requests are sent to the radar scheduler independently. That is, each task makes look requests based only on its own requirements. The role of the radar scheduler is to receive all look requests and formulate a schedule for the radar, under the constraint that at any given time, the radar only executes one look. The radar scheduler must decide whether or not to schedule the look request. For example, if two look requests which start at the same time are received, the scheduler must decide whether to alter the start times of one or both looks or to not schedule one of the looks.

The three basic radar functions are assumed to have following priorities.

1. Weapon control (highest priority)
2. Tracking (medium priority)
3. Surveillance (lowest priority)

Within each function, tasks may have different priorities. For example, tracking task A may have a higher priority than tracking task B if the target associated with task A is thought to be more of a threat than the target associated with task B. Since weapon control is the highest priority function, if any weapon control look requests are received, the radar scheduler will usually schedule any such look requests and cancel or delay all pending look requests from surveillance or tracking tasks. For the purpose of developing the proposed sequential scheduler, attention will be restricted to the tracking and surveillance functions. That is, the weapon control function will not be considered in this work.

In this study, the scheduler receives all look requests and formulates the schedule for a window of a fixed length of time. After the schedule has been carried out by the radar, the scheduler then formulates the schedule for the next window. Choosing shorter or longer windows have different advantages. If a shorter window is utilized, then newer look requests may be considered more quickly. Furthermore, the use of a shorter window results in a smaller number of look requests which must be processed at one time by the scheduler. If a longer window is utilized, a larger number of look requests must be processed, but the scheduling is carried out less frequently. The choice of window length is a balance between the need to rapidly consider new look requests and the desire to formulate schedules less frequently.

By assumption, tracking looks have a higher priority than surveillance looks. Tracking and surveillance looks also differ in a number of ways. At any given time, the radar maintains a variable number of tracks. Depending on the dynamics of the target, each track may need to be updated more or less frequently. Therefore, tracking looks are by nature variable in quantity. It is possible for the scheduler to receive no tracking look requests for a period of time. It is also possible for the scheduler to receive so many tracking look requests in a window that some requests must be dropped. Furthermore, a tracking look request is sensitive to its scheduled time. As the time between track updates increases, the uncertainty in the predicted position of the target increases. This results in the radar having to search a larger region to detect the target, which increases the length of the radar look. If a long period of time elapses between track updates, then the track is lost. A long delay in scheduling a tracking look can result in the elimination of the associated task.

On the other hand, surveillance is a persistent function, so that surveillance look requests are always present. The length of time required to carry out a surveillance look is typically fixed in length, since each look request depends on the area to be monitored but not on the dynamics of a potential target. Surveillance look requests may be thought of as a queue. If the radar is not occupied with tracking looks, then the next surveillance look request

in the queue should be scheduled. It is desirable to carry out surveillance of a particular area as often as possible. However, even if a long period of time has elapsed since the last surveillance look, carrying out a new surveillance look is always beneficial.

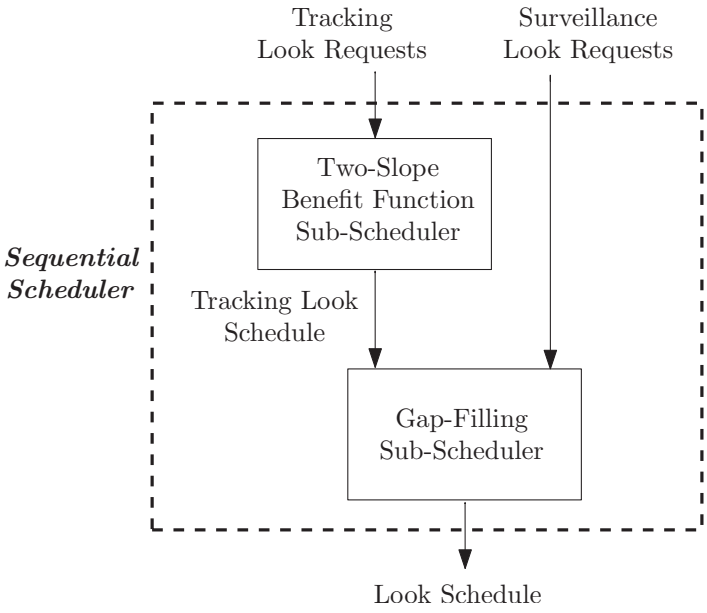

Figure 1: *Illustration of the Sequential Scheduler.*

Based on the differing priorities and nature of the tracking and surveillance looks, a method called the Sequential Scheduler is proposed. The scheduler, which is shown in Figure 1, consists of two components. The Two-Slope Benefit Function Sub-Scheduler generates a schedule of tracking look requests. The Gap-Filling Sub-Scheduler considers the tracking look schedule and schedules surveillance look requests in any remaining idle intervals within the window. The TSBF Sub-Scheduler is described in Section 3. The Gap-Filling Sub-Scheduler is described in Section 4. Scheduling tracking look requests first ensures that higher priority tracking looks are scheduled before lower priority surveillance looks. The surveillance looks are then scheduled to occupy as much of the radar window as possible.

This scheduler adaptively schedules look requests of arbitrary lengths, which are specified by the tracker or surveillance manager. Furthermore, the Sequential Scheduler is able to accommodate adaptive update rates, since desired start times may be chosen arbitrarily.

Note that the Sequential Scheduler does not place any constraints on the total time scheduled for tracking requests. It is possible to place a constraint on the Sequential Scheduler so that a maximum percentage of the window is devoted to tracking looks. Once the total length of scheduled tracking looks reached the maximum, surveillance looks would then be scheduled in the remaining radar time line. In this paper, no such constraints are placed

on the TSBF Sub-Scheduler.

3 Two-slope benefit function sub-scheduler for tracking looks

This section describes the first component of the Sequential Scheduler, the Two-Slope Benefit Function Sub-Scheduler. The TSBF Sub-Scheduler receives tracking look requests and each look request is either selected with a start time or dropped.

3.1 Preliminaries

Consider a time window $[T_1, T_2]$. Associated with this window, the sub-scheduler receives P tracking look requests $L_1, L_2, ..., L_P$. Each look request L_n has the look parameters

- l_n, the time required to complete the look, in seconds,
- t_n^*, the desired start time,
- s_n, the earliest start time,
- u_n, the latest start time.
- B_n^*, peak benefit,
- δ_n, slope for early scheduling,
- Δ_n, slope for late scheduling.

The look parameters satisfy $s_n \leq t_n^* \leq u_n$. The interval $[s_n, u_n]$ is called the scheduling interval for L_n. All look requests are assumed to have a scheduling interval which lies within $[T_1, T_2]$. The slopes for early and late scheduling are restricted to $0 < \delta_n < \infty$ and $0 < \Delta_n < \infty$. If a look is scheduled, the start time t_n must satisfy $s_n \leq t_n \leq u_n$. The look request L_n is associated with a benefit function $B_n(t_n)$, which measures the benefit of selecting start time t_n. The benefit function is a two-slope function, which is given by

$$B_n(t_n) = B_n^* + c_n(t_n - t_n^*), \tag{1}$$

where

$$c_n = \begin{cases} \delta_n, & s_n \leq t_n \leq t_n^* \\ -\Delta_n, & t_n^* < t_n \leq u_n \end{cases} . \tag{2}$$

A two-slope benefit function is illustrated in Figure 2.

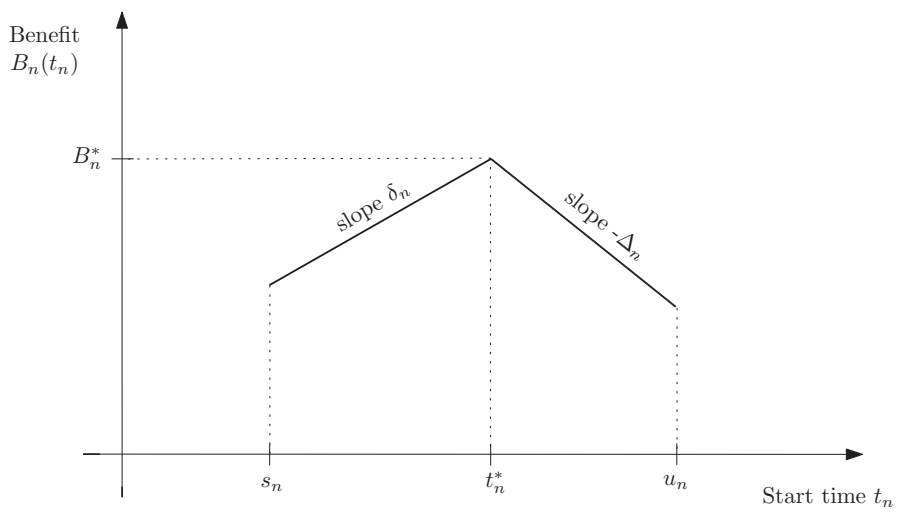

Figure 2: *A two-slope benefit function.*

Without loss of generality, the looks are ordered so that

$$t_1^* \leq t_2^* \leq t_3^* \leq \cdots \leq t_P^*.$$

This ordering is a labeling convention and does not restrict the arbitrary choice of any of the look parameters. Note that it is not necessarily true that $s_n \leq s_{n+1}$ or $u_n \leq u_{n+1}$ for any $n = 1, ..., P-1$, since the scheduling intervals for the looks may have different lengths.

Definition: A start time t_n for look L_n is a **viable start time** if $s_n \leq t_n \leq u_n$.

If the start time for look L_n is t_n, then the look ends at time $t_n + l_n$. Because only one look may be executed at a time, the start time for the next scheduled look must be no earlier than $t_n + l_n$. This leads to the following definition.

Definition: Consider a set of looks $L_1, ..., L_P$ with ordered start times $t_1 < t_2 < t_3 < \cdots < t_P$. The set $L_1, ..., L_P$ is a **viable set** if

$$t_n + l_n \leq t_{n+1}, \text{for all } n = 1, 2, ..., P-1.$$

Due to the large number of parameters used by the TSBF Sub-Scheduler, a list of parameters is specified in Table 1

Table 1: *List of parameters for the TSBF Sub-Scheduler.*

	Parameter	Description
Look Parameters (Sect. 3.1)	L_n	look request
	l_n	time to complete look
	t_n^*	desired start time
	s_n	earliest start time
	u_n	latest start time
	B_n^*	peak benefit
	δ_n	slope for early scheduling
	Δ_n	slope for late scheduling
	t_n	start time
	$B_n(t_n)$	benefit function
Metrics Calculation (Sect. 3.4)	t_n'	conditional earliest start time
	E_n	maximum delay within a sequence
	Q	number of sequences
	D_q	starting look for sequence q
	G_q	maximum delay between sequences
Start Time Assignment (Sect. 3.6)	α_n	delay from earliest conditional start time
	β_n	allowable difference between delays
	$\overline{B}(t_1,...,t_N)$	total benefit
	v_n	auxiliary variable for delay within a sequence
	w_n	auxiliary variable for delay between sequences
	x_n	additional variable for piecewise linear objective function
	$\widehat{t_n}$	optimal start time

3.2 Radar loading of tracking looks within the time window

For a set of tracking look requests $L_1, ..., L_P$ with lengths $l_1, ..., l_P$, define

$$\bar{l} = \sum_{n=1}^{P} l_n.$$

This is the minimum total time required to complete all looks. Also define

$$\tau = \max_n (u_n + l_n) - \min_n s_n.$$

This is the maximum amount of radar time line available for the given set of looks. Note that in many cases, it may not be possible for all looks to be completed in the time τ, because this would require that multiple looks be executed by the radar simultaneously. The quantity τ is defined to facilitate a definition of radar loading.

Definition: The radar is **underloaded** during the time window if $\bar{l} \leq \tau$.

An underloaded radar may be able to schedule all P looks in τ seconds. However, it may be able to schedule all looks only by starting some of them at times other than the desired start times. Note that being underloaded is a necessary, but not sufficient, condition for the radar to execute all looks.

Definition: The radar is **overloaded** during the time window if $\bar{l} > \tau$.

An overloaded radar cannot schedule all P looks, and some looks must be dropped. With these definitions of loading, a set of looks $\{L_n\}$ can be categorized into one of three conditions, as follows.

Definition: A set of look requests $L_1, ..., L_P$ is in **Condition I** if the radar is underloaded and

$$t_n^* + l_n \leq t_{n+1}^*, \text{for all } n = 1, 2, ..., P-1. \tag{3}$$

That is, every look can be scheduled at its desired start time. This case is straight-forward and does not require the sub-scheduler to make any decisions about shifting start times away from desired start times or dropping looks.

Definition: A set of look requests $L_1, ..., L_P$ is in **Condition II** if the radar is underloaded and (3) does not hold.

For a set of look requests in Condition II, it may be possible for all looks to be scheduled, but only if at least one of the looks does not start at its desired start time. It may also be the case that some looks must be dropped.

Definition: A set of look requests $L_1, ..., L_P$ is in **Condition III** if the radar is overloaded.

In this case, the radar is overloaded, and some looks must be dropped. The sub-scheduler must also select start times for the looks.

For Condition I, all looks can be scheduled at their desired start times, so scheduling is trivial. For Conditions II and III, the sub-scheduler may have to decide which looks to drop, and must schedule each of the looks. The TSBF Sub-Scheduler is used for look requests which are in Condition II or III.

3.3 Sub-scheduler overview

The input to the TSBF Sub-Scheduler is a set of P tracking look requests. The output of the sub-scheduler is a viable subset of N looks, where $N \leq P$, and start times for each of the N looks. The viable subset may be the entire set of P look requests. The main components of the TSBF Sub-Scheduler are shown in Figure 3.

A radar scheduler must decide whether or not to schedule a look request, and if the look is to be scheduled, select a start time. The TSBF Sub-Scheduler carries out these two functions separately, by first deciding which looks are to be scheduled, and then selecting start times for the resulting subset.

Recall that the desired start times of the look requests are ordered so that $t_1^* \leq t_2^* \leq t_3^* \leq \cdots \leq t_P^*$. The TSBF Sub-Scheduler assumes that looks are scheduled in sequence, so that $t_i < t_j$ for $i < j$. If all looks can be scheduled at their desired start time, then this assumption is sound. Although in some special cases it may be advantageous to schedule the looks out of sequence, the consideration of different ordering combinations increases the computational complexity of the sub-scheduler. It is likely that scheduling the looks in sequence is advantageous in most cases.

The set of P look requests $L_1, ..., L_P$ are first processed to produce a set of look metrics. Details of the Metrics Calculation Algorithm are given in Section 3.4. The resulting metrics are used to determine if the look requests are viable. If the set of look requests is viable, then the viable set is sent to the Start Time Assignment Algorithm. If the set of look requests is not viable, then the set is subject to the Look Down-Selection Algorithm, which produces a viable subset of looks. This subset will necessarily be a strict subset, since one or more look requests will need to be dropped. Details of the Look Down-Selection Algorithm are presented in Section 3.5. The Start Time Assignment Algorithm produces a set of start times which maximize the total benefit of the viable set of looks. Details are given in Section 3.6.

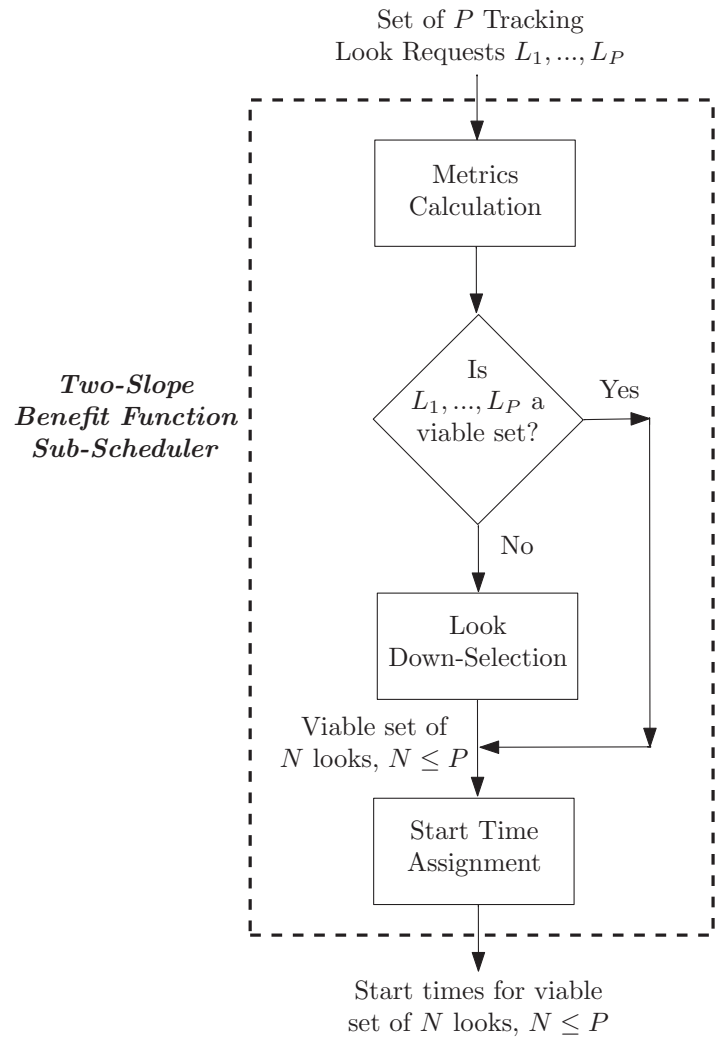

Figure 3: *Overview of the Two-Slope Benefit Function Sub-Scheduler.*

3.4 Metrics calculation

The Metrics Calculation Algorithm is the first stage of the TSBF Sub-Scheduler. This algorithm is also used within the Look Down-Selection Algorithm.

Let $L_1,...,L_P$ be the input set of look requests. The selected start times of the looks are assumed to be ordered so that $t_1 < t_2 < t_3 < \cdots < t_P$. A number of metrics are calculated, including $\{t'_n\}_{n=1}^{P}$, $\{E_n\}_{n=1}^{P}$, an integer Q where $1 \leq Q \leq N$, $\{D_q\}_{q=1}^{Q}$, and $\{G_q\}_{q=1}^{Q-1}$ when $Q \geq 2$. The metrics $\{t'_n\}_1^P$ are the earliest available start times for each look, given that each previous look has been scheduled at its earliest available start time. The start times t'_n will be called the conditional earliest start times. The metrics $\{E_n\}_{n=1}^{P}$, Q, $\{D_q\}_{q=1}^{Q}$, and $\{G_q\}_{q=1}^{Q-1}$ quantify the maximum delay that can be applied to each look while remaining viable.

The metrics $\{E_n\}_{n=1}^{P}$ will be utilized to determine whether the set of look requests is viable. For a viable set of looks, all of the metrics will be used to assign the start times.

The metrics are calculated as follows.

1. Let $t'_1 = s_1, E_1 = u_1 - s_1, q = 1, D_q = 1$, and $n = 2$.
2. Let $t'_n = \max(s_n, t'_{n-1} + l_{n-1})$ and $E_n = u_n - t'_n$. If $s_n > t'_{n-1} + l_{n-1}$, then let $G_q = s_n - t'_{n-1} - l_{n-1}, q = q + 1$, and $D_q = n$.
3. If $n = P$ then $Q = q$ and stop. Otherwise, let $n = n + 1$ and go to step 2.

The metrics $\{t'_n\}$ are a set of start times which satisfy $t'_n \geq s_n$ for all n. However, $\{t'_n\}$ do not necessarily satisfy $t'_n \leq u_n$ for all n. An interpretation for $\{t'_n\}$ is as follows. The start time t'_1 is the earliest possible start time for L_1. For this choice of start time, L_1 ends at time $t'_1 + l_1$. The start time t'_2 is the earliest possible start time for L_2 given that L_1 started at time t'_1. If $t'_1 + l_1 \geq s_2$, then L_2 can start at time $t'_1 + l_1$, but if $t'_1 + l_1 < s_2$, then L_2 must wait until time s_2 to start, since the start time must satisfy $t'_2 \geq s_2$. For L_n the start time t'_n is the earliest possible start time given that L_m started at time t'_m for $m = 1,...,n-1$. The start times $t'_1,...,t'_P$ are chosen without considering the latest start times.

For each n, E_n is the difference between u_n and t'_n; that is, the difference between the latest start time and the conditional earliest start time. The definition of $\{E_n\}_{n=1}^{P}$ results in a test for the viability of the set of look requests.

Test for Viability: If $E_n \geq 0$ for $n = 1,...,P$, then $\{L_1,...,L_P\}$ is viable set and $\{t'_n\}_{n=1}^{P}$ is a set of viable start times.

To prove this, note that by definition $t'_n \geq s_n$ for $n = 1,...,P$ and $t'_{n+1} \geq t'_n + l_n$ for $n = 1,...P-1$. If $E_n \geq 0$ for all n, then $u_n \geq t'_n$ for all n, which shows that $t'_1,...,t'_P$ are viable start times. Therefore, $L_1,...,L_P$ is a viable set.

The Metrics Calculation Algorithm partitions the look requests into Q sequences. The first look request in sequence q is denoted D_q, and the first look of the first sequence is look L_1. For every look request in a sequence, except for the last look, the end time of the look request equals the start time of the next look request. For all sequences except the last sequence, the difference between the end time of the last look of sequence q and the start time of the next sequence is G_q. Because Q is the last sequence, G_Q is undefined.

Let $E_n < 0$ for a given n, and let q be the sequence containing look L_n. An explanation of the condition $E_n < 0$ is as follows. Since L_n is in sequence q, it is known that

$$t'_n = s_{D_q} + \sum_{i=D_q}^{n-1} l_i,$$

and that $u_n < t'_n$. If all looks prior to L_n in sequence q are scheduled at their earliest possible time, then a viable start time for L_n cannot be selected. In order for L_n to be scheduled at a viable start time, one or more of the prior looks in sequence q must be dropped.

The Metrics Calculation Algorithm serves two purposes in the TSBF Sub-Scheduler. First, calculation of the metrics $\{E_n\}$ lead to a test for the viability of the set of look requests. Second, the look metrics are used by the Start Time Assignment Algorithm to compute the start times that maximize total benefit. It is evident that the Metrics Calculation Algorithm has low computational complexity.

3.5 Look down-selection

The look metrics lead to a test for the viability of a set of look requests: if $E_n \geq 0$ for all n, then the set is viable. If the set is not viable, then the Look Down-Selection Algorithm drops one or more look requests and produces a viable set of looks.

A flowchart for the Look Down-Selection Algorithm is given in Figure 4. The look requests are sorted by peak benefit in descending order. The sorted look requests are labeled $\{\Lambda_n\}_{n=1}^{P}$. The goal of the down-selection is to select a viable subset C of $\{\Lambda_n\}_{n=1}^{P}$. Starting with the look request with the largest peak benefit, subsequent look requests are added one at a time to the set. The resulting new set is sent to the Metrics Calculation Algorithm. If this new set is viable, then the most recent addition to the set is kept. Otherwise, the most recent addition is dropped. This process continues until all look requests have been considered. The look with the largest peak benefit, Λ_1, will always be included in C. Note that at least one look request will be dropped, because the original set of look requests is not viable.

As look requests are added to the set, the metrics are calculated to determine viability. If a set is not viable, dropping the most recent addition to the set is advantageous for two reasons. First, the most recent addition is the look with the smallest peak benefit which has

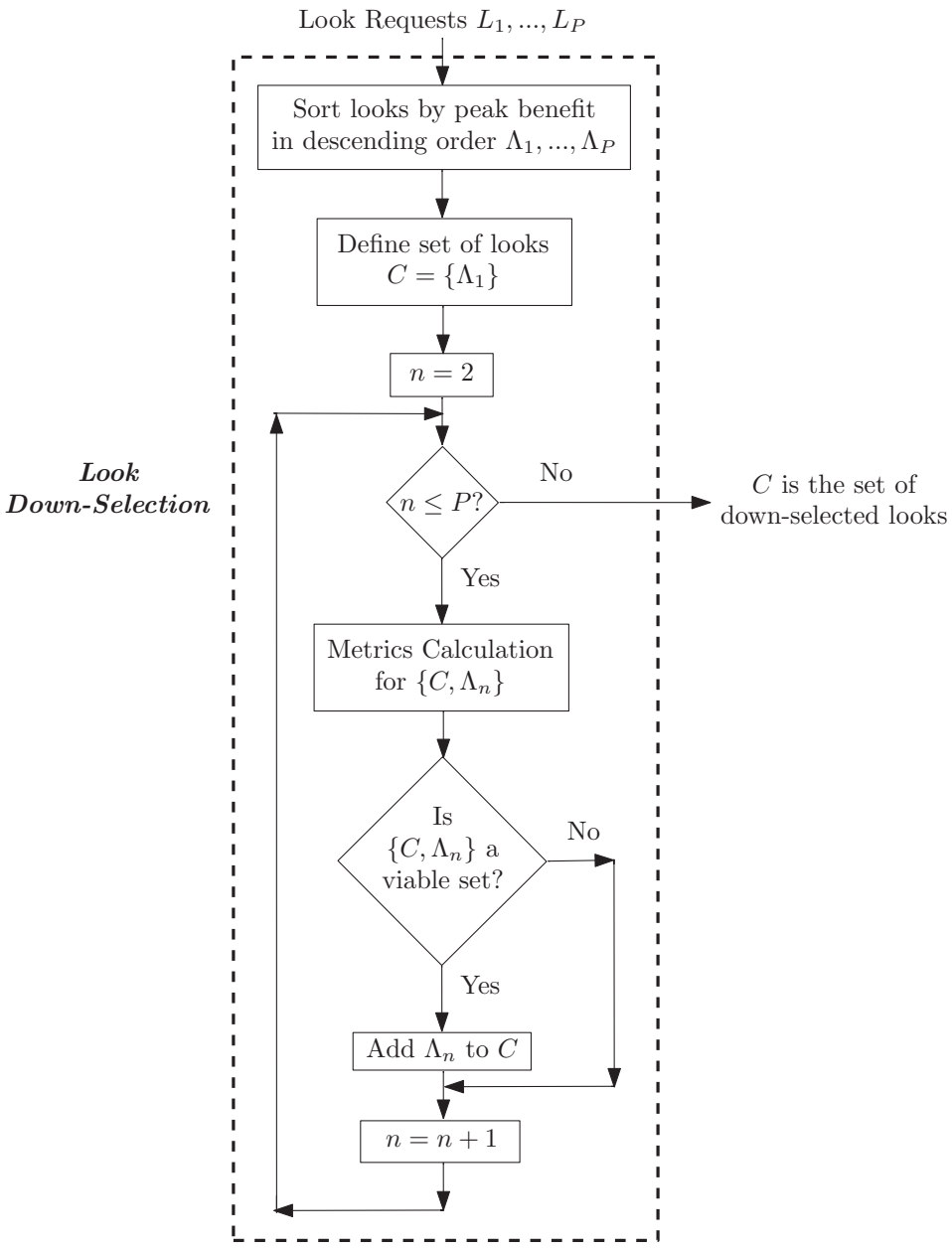

Figure 4: *The Look Down-Selection Algorithm.*

been considered at that time. Second, dropping the most recent addition results in a viable set.

The relative peak benefit of a look request plays an important role in whether the look is down-selected. The look request with the largest peak benefit is always down-selected. To be down-selected, any other look request, together with all look requests with larger peak benefit, must be a viable set. For look requests with larger peak benefit, there are fewer other look requests in the set, which enhances the probability of being down-selected. Look requests with longer scheduling intervals also have a higher probability of being down-selected.

The Look Down-Selection Algorithm produces a viable subset C of the original set of look requests $L_1, ..., L_P$. The Metrics Calculation Algorithm is executed $P-1$ times by the Look Down-Selection Algorithm. As noted in Section 3.4, the computational requirements for calculating the look metrics are light.

3.6 Start time assignment

The input to the Start Time Assignment Algorithm is a viable set of looks. If the original set of look requests was viable, then the input set is the original set of looks. If not, then the input set of looks is the viable subset generated by the Look Down-Selection Algorithm. The input set of looks is $\{L_n\}_{n=1}^{N}$. The metrics $\{t'_n\}_{n=1}^{N}$, $\{E_n\}_{n=1}^{N}$, Q, $\{D_q\}_{q=1}^{Q}$, and $\{G_q\}_{q=1}^{Q-1}$ (for $Q \geq 2$) were calculated for the input set of looks by the Metrics Calculation Algorithm. The Start Time Assignment Algorithm schedules all looks in the input set.

Because all looks are scheduled, the start time t'_n is the earliest possible viable start time for L_n. All viable start times can be expressed as

$$t_n = t'_n + \alpha_n, n = 1, ..., N,$$

where α_n is the delay from the conditional earliest start time and is subject to the constraints

$$0 \leq \alpha_n \leq E_n, n = 1, ..., N, \tag{4}$$

$$\alpha_n \leq \alpha_{n+1} + \beta_n, n = 1, ..., N-1, \tag{5}$$

where

$$\beta_n = \begin{cases} 0, & \text{if } L_n \text{ and } L_{n+1} \text{ are in the same sequence} \\ G_q, & \text{if } L_n \text{ is in sequence } q \text{ and } L_{n+1} \text{ is in sequence } q+1 \end{cases}. \tag{6}$$

The Start Time Assignment Algorithm calculates viable start times to maximize the total benefit, which is given by

$$\overline{B}(t_1, ..., t_N) = \sum_{n=1}^{N} B_n(t_n). \tag{7}$$

It will be shown that the benefit functions can be expressed in the form

$$B_n(t_n) = B_n(t'_n) + f_n(\alpha_n). \tag{8}$$

Total benefit can then be expressed as

$$\overline{B}(t_1,...,t_N) = \sum_{n=1}^{N} B_n(t_n) = \sum_{n=1}^{N} \left[B_n(t'_n) + f_n(\alpha_n) \right].$$

Therefore choosing viable start times which maximize (7) is equivalent to choosing $\{\alpha_n\}_1^N$ to maximize

$$\sum_{n=1}^{N} f_n(\alpha_n), \tag{9}$$

subject to the linear inequality constraints (4) and (5). Furthermore, if $f_n(\alpha_n)$ is linear in α_n, then the maximisation problem is a linear program and can be solved by applying the simplex method for linear programming [10].

It is shown that a benefit function can be expressed in the form given by (8). First, consider the case where $t'_n > t^*_n$.

$$\begin{aligned} B_n(t_n) &= B_n(t'_n + \alpha_n) \\ &= B^*_n - \Delta_n(\alpha_n - t^*_n + t'_n) \\ &= B_n(t'_n) - \Delta_n \alpha_n \end{aligned}$$

Therefore $f_n(\alpha_n)$ can be expressed as a linear equation

$$f_n(\alpha_n) = -\Delta_n \alpha_n.$$

Next, consider the case where $t'_n \leq t^*_n$. It is seen that

$$\begin{aligned} B_n(t_n) &= B_n(t'_n + \alpha_n) \\ &= \begin{cases} B^*_n - \delta_n(t^*_n - t'_n) + \delta_n \alpha_n, & 0 \leq \alpha_n < t^*_n - t'_n \\ B^*_n - \delta_n(t^*_n - t'_n) + \delta_n(t^*_n - t'_n) - \Delta_n(\alpha_n - t^*_n + t'_n), & t^*_n - t'_n \leq \alpha_n \leq E_n \end{cases} \\ &= B_n(t'_n) + f_n(\alpha_n), \end{aligned}$$

where the function $f_n(\alpha_n)$ is given by

$$f_n(\alpha_n) = \begin{cases} \delta_n \alpha_n, & 0 \leq \alpha_n < t^*_n - t'_n \\ \delta_n(t^*_n - t'_n) - \Delta_n(\alpha_n - t^*_n + t'_n), & t^*_n - t'_n \leq \alpha_n \leq E_n \end{cases}. \tag{10}$$

In this case, $f_n(\alpha_n)$ is a piecewise-linear equation. The traditional simplex method for linear programming requires that the objective function be linear. Simplex methods for piecewise-linear objective functions have been developed [11], [12]. The approach taken

here is to convert the piecewise-linear objective function to an equivalent linear objective function through the use of additional variables [13]. This then allows for the application of the traditional simplex method. Returning to (10), when $t_n^* - t_n' \leq \alpha_n \leq E_n$,

$$\begin{aligned} f_n(\alpha_n) &= \delta_n(t_n^* - t_n') - \Delta_n(\alpha_n - t_n^* + t_n') \\ &= \delta_n\alpha_n - (\delta_n + \Delta_n)(\alpha_n - t_n^* + t_n') \end{aligned}$$

Therefore the function $f_n(\alpha_n)$ is given by

$$f_n(\alpha_n) = \begin{cases} \delta_n\alpha_n, & 0 \leq \alpha_n < t_n^* - t_n' \\ \delta_n\alpha_n - (\delta_n + \Delta_n)(\alpha_n - t_n^* + t_n'), & t_n^* - t_n' \leq \alpha_n \leq E_n \end{cases}$$

For $0 \leq \alpha_n \leq E_n$, $f_n(\alpha_n)$ can be expressed as the linear equation

$$f_n(\alpha_n) = \delta_n\alpha_n - (\delta_n + \Delta_n)\phi_n,$$

where

$$\begin{aligned} \phi_n &= \begin{cases} 0, & 0 \leq \alpha_n < t_n^* - t_n' \\ \alpha_n - t_n^* + t_n', & t_n^* - t_n' \leq \alpha_n \leq E_n \end{cases} \\ &= \max(0, \alpha_n - t_n^* + t_n'). \end{aligned}$$

The optimisation problem is thus a linear program, which is summarized as follows. Let N_E be the set of all n for which $t_n' \leq t_n^*$. Choose $\{\alpha_n\}_1^N$ and $\{\phi_n\}_{n \in N_E}$ to maximize (9), where

$$f_n(\alpha_n) = \begin{cases} \delta_n\alpha_n - (\delta_n + \Delta_n)\phi_n, & \text{if } t_n' \leq t_n^* \\ -\Delta_n\alpha_n, & \text{if } t_n' > t_n^* \end{cases}, \tag{11}$$

subject to the constraints

$$v_n = E_n - \alpha_n, n = 1, ..., N, \tag{12}$$

$$w_n = \alpha_{n+1} - \alpha_n + \beta_n, n = 1, ..., N-1, \tag{13}$$

$$x_n = \phi_n - \alpha_n + t_n^* - t_n', n \in N_E, \tag{14}$$

where β_n is given by (6) and $\alpha_n, \phi_n, v_n, w_n, x_n \geq 0$. The simplex method is used to compute the values for the variables $\alpha_n, \phi_n, v_n, w_n, x_n$ which maximize (9). Details of applying the simplex method to this optimisation problem are given in Annex A. The processing time for one hundred look requests in on the order of one second, depending on the look parameters.

Let $\{\widehat{\alpha}_n\}_1^N$ and $\{\widehat{\phi}_n\}_{n \in N_E}$ be the set of variables which maximize (9). The start times that maximize the total benefit are then given by

$$\widehat{t_n} = t_n' + \widehat{\alpha}_n, n = 1, ..., N.$$

There are noteworthy special cases of this optimisation problem. First, consider the case where $t'_n \geq t^*_n$ for all n. Examination of (11) shows that (9) is maximized by selecting $\alpha_n = 0$ for all n. The simplex method does not need to be carried out, and the start times $\{t'_n\}_{n=1}^N$ are optimal. Note that if $s_n = t^*_n$ for all n, then by definition $t'_n \geq t^*_n$ for all n.

Another special case occurs when $t^*_n = u_n$ for all n. In this case, the optimisation problem simplifies to choosing $\{\alpha_n\}_1^N$ to maximize (9), where

$$f_n(\alpha_n) = \delta_n \alpha_n,$$

subject to the constraints

$$\begin{aligned} v_n &= E_n - \alpha_n, n = 1, ..., N, \\ w_n &= \alpha_{n+1} - \alpha_n + \beta_n, n = 1, ..., N-1, \end{aligned}$$

where β_n is given by (6) and $\alpha_n, v_n, w_n \geq 0$. In this case, the benefit function given by (1) is linear, as opposed to piecewise-linear for the general case. Therefore, the variables ϕ_n and x_n do not need to be introduced to convert the optimisation problem to a linear program.

The benefit function was defined as a two-slope function, as given by (1) and (2). Benefit functions with other forms can also be formulated as shown in Annex B. The Metric Calculation and Look Down-Selection Algorithms are not dependent on the form of the benefit function. Only the optimisation in the Start Time Assignment Algorithm is dependent on the form of the benefit function.

3.7 Selection of look parameters

For the TSBF Sub-Scheduler, each look request has a set of look parameters which must be chosen. The choice of certain parameters affects whether the look request will be down-selected and if the look request is down-selected, how close the start time will be to the desired start time.

If the original set of look requests is viable, then down-selection is not required. However, if the original set is not viable, then the peak benefit B^*_n plays an important role in the down-selection process. Because the look requests are ordered in descending order by peak benefit, look requests with larger peak benefits generally have a better chance of being down-selected. This is due to the fact that when it is being considered for inclusion in the set C, there are fewer other look requests, which together with the look request under consideration, may result in a set that is not viable.

Once a set of viable look requests has been generated, the choice of slopes for early and late scheduling, δ_n and Δ_n affect how close the look will be scheduled to its desired start time. If a look has larger values of δ_n and Δ_n relative to the other looks, then the look under consideration will be scheduled closer to its desired start time. This assumes that there is

flexibility in choosing the start times for the viable set of looks. In cases where the set of looks is almost fully loaded, there may be little flexibility in choosing the start times.

Therefore, there are two distinct types of priority for each look request. A larger peak benefit, B_n^*, increases the likelihood that a look request will be down-selected. Larger values of δ_n and Δ_n increase the likelihood that a look will be scheduled closer to its desired start time. This is distinct from the traditional notion of task priority, which considers a single measure for priority. Further investigation will need to be carried out to study methods for selecting B_n^*, δ_n and Δ_n. Previous work on task prioritisation would be a useful starting point for such a study. In particular, approaches based on fuzzy logic [2],[14] and neural networks [15] have produced methods for prioritizing target tracks.

The TSBF Sub-Scheduler also requires that the scheduling interval and desired start time be selected. Tracking update rates, which can be used to compute the desired start time have been studied in [16], [17] and [18]. A formula for the latest start time has also been derived in [16]. The specification of the scheduling interval and its effect on tracking performance is another area that requires further study.

3.8 Summary

The TSBF Sub-Scheduler receives tracking look requests and generates the start times for a viable subset of tracking looks. If necessary, down-selection is carried out by a process which favours higher peak benefit look requests. Start times are chosen to maximize the total benefit of the viable look requests. It is shown that the maximisation problem can be expressed as a linear program, which allows for the application of the simplex method.

4 Gap-filling sub-scheduler for surveillance looks

Inputs to the Gap-Filling Sub-Scheduler are the tracking look schedule generated by the TSBF Sub-Scheduler and a set of surveillance look requests. If the entire window is occupied with tracking looks, then the GF Sub-Scheduler does nothing, and the track schedule is the final schedule for the Sequential Scheduler. However, if there are any idle time intervals in the window, then the GF Sub-Scheduler attempts to schedule surveillance looks in the idle time intervals. The surveillance look requests are assumed to be organized as a queue, as explained in Section 4.1. Details of the GF Sub-Scheduler are presented in Section 4.2.

4.1 Queue management for surveillance look requests

The surveillance function consists of M tasks, where each task involves the monitoring of a region in space which is defined by a distinct beam position or several beam positions of the radar. Each task generates a look request which is sent to the sub-scheduler. When the look from Task m is carried out, a new look request from that same task is generated. Therefore, the sub-scheduler is always in possession of a single look request from each task. Let the look request from Task m be labeled λ_m, where $m = 1, ..., M$. For this study, it is assumed that the length of each surveillance look is d for all looks. In general, the dwell time for different looks may have different lengths. Unlike with a tracking look request, there is no scheduling interval associated with a surveillance look request. There is no earliest start time, because from the point of view of Task m, persistent surveillance of beam position m is of maximum benefit. There is also no latest start time, because even if a long period of time has elapsed since beam position m was last monitored, it is beneficial to the radar to schedule λ_m, because the surveillance of the region associated with Task m will enhance the surveillance picture.

The sub-scheduler is always in receipt of M surveillance look requests. Associated with each look request is the elapsed time ε_m, which computes the time which has elapsed since the last time a look from Task m was carried out. Allow the requests to be organized in a queue, so that the sub-scheduler chooses the first look request in the queue. Ordering the look requests in the queue will determine how the look requests are scheduled. Two different cases are considered: one where all look requests have equal priority, and one where the look requests have different priorities.

4.1.1 Equal priority looks

When all M looks have equal priority, then the look requests form a first-in, first-out (FIFO) queue. After each look is chosen from the top of the queue and scheduled, it is inserted at the bottom of the queue. The FIFO queue is equivalent to ordering the look requests by ε_m in descending order. The look request with the largest ε_m is the next one to be scheduled.

Consider an example with six looks, where each look has equal priority. Let the current time be time zero. Table 2 shows the queue at time zero. The queue is ordered in descending order of elapsed time ε_m. If a surveillance look is to be scheduled, look λ_3 will be chosen from the top of the queue. Now assume that a surveillance look is not scheduled at time zero nor at any time between time zero and time t. Although the values of ε_m will all increase by t between time zero and time t, the ordering of the looks in the queue will not change. For this equal priority example, regardless of when the next look is scheduled, λ_3 will be the next look chosen.

Table 2: *Look queue with equal priority looks, at time zero.*

Look	Elapsed Time ε_m (sec.)
λ_3	6.7
λ_4	6.5
λ_5	6.1
λ_6	6.0
λ_1	5.3
λ_2	4.5

4.1.2 Looks with unequal priorities

Now consider the case where the looks have different priorities. In this case, each look has a priority value which is specified by a time ω_m in seconds. When the sub-scheduler selects a look to be scheduled, all looks with the smallest value of ω_m and for which $\omega_m \leq \varepsilon_m$ are placed at the top of the queue. If there is more than one such look, then the looks are ordered by ε_m in descending order. This is a generalisation of the case where all looks have equal priority, since choosing $\omega_m = \infty$ for all m results in a FIFO queue. In this scheme, higher priority looks will have a smaller value of ω_m, which results in look λ_m being scheduled more often.

Table 3: *Look queue with unequal priorities, at time zero.*

Look	Priority ω_m (sec.)	Elapsed Time ε_m (sec.)
λ_6	*5.5*	6.0
λ_4	*6.0*	6.5
λ_3	∞	6.7
λ_5	∞	6.1
λ_1	∞	5.3
λ_2	*5.0*	4.5

Returning to the example with six looks, the elapsed times are assumed to be the same as in the previous example with equal priority looks. Let the current time be time zero. In this case, each of the looks is assigned a priority value, as shown in Table 3. Looks λ_4 and λ_6 have an elapsed time which is greater than their priority, so they are placed ahead of the other looks in the queue. Look λ_6 has a smaller elapsed time than look λ_4, but λ_6 has a smaller value of ω_m so it is placed ahead of λ_4 in the queue. Look λ_2 has a priority value of $\omega_2 = 4.0$, but its elapsed time is less than its priority, so it is not advanced in the queue. At time zero, if a surveillance look is to be scheduled, look λ_6 will be chosen from the top of the queue. If a look is not scheduled at time zero, but is chosen between time zero and time

0.5 seconds, then look λ_6 will still be scheduled, since it will be at the top of the queue.

Assume that no surveillance looks are scheduled at time zero or at any time between time zero and time 0.5. At time 0.5, the elapsed times for all looks will have increased by 0.5. Since $\omega_2 = \varepsilon_2$ and ω_2 is the smallest priority value, look λ_2 will move to the top of the queue, as shown in Table 4. Thus, it is seen that the ordering in the queue can change with time, depending on the values of ω_m in relation to the elapsed time of each of the looks.

Table 4: *Look queue with unequal priorities, at time 0.5.*

Look	Priority ω_m (sec.)	Elapsed Time ε_m (sec.)
λ_2	*5.0*	5.0
λ_6	*5.5*	6.5
λ_4	*6.0*	7.0
λ_3	∞	7.2
λ_5	∞	6.6
λ_1	∞	5.8

4.2 Sub-scheduler operation

The Gap-Filling Sub-Scheduler starts with the tracking look schedule that was generated by the TSBF Sub-Scheduler and considers all intervals in the window during which no tracking looks have been scheduled. The goal is to schedule as many surveillance looks as possible in each idle interval. Let I be the length of an idle interval. If $I < d$, then no surveillance looks are scheduled. Otherwise, k looks are scheduled in the interval, where k satisfies

$$kd \leq I < (k+1)d.$$

When one or more looks are to be scheduled, the GF Sub-Scheduler computes the state of the queue at that time, and the looks at the top of the queue are scheduled.

5 Scheduling examples

The Sequential Scheduler is a general scheduling technique that can be applied to tracking and surveillance look requests with arbitrary parameters. In this section, several examples are considered which demonstrate some characteristics of the scheduler. In each example, the scheduler receives a number of surveillance and/or look requests and produces a schedule for these requests for a single window. Although only a single window is considered in these examples, an operational scheduler would schedule looks requests sequentially for consecutive windows.

For all examples, the start time of the window is 0 ms. The nominal end time of the window is 300 ms. The scheduler will schedule all looks that start before the nominal end time. The actual end time of the window is when the last look is completed. There are 20 surveillance look requests, all with equal priority. The queue of surveillance look requests is given in descending order by $\lambda_1, \lambda_2, ..., \lambda_{20}$. The length of each surveillance look is d=15 ms. In the examples presented, $\delta_n = \Delta_n$ for all tracking look requests; however, in general δ_n and Δ_n can be selected independently of each other. Recall that a list of parameters for the TSBF Sub-Scheduler is provided in Table 1.

5.1 Example 1: No tracking looks

In this example, no tracking look requests are received by the scheduler. As a result, surveillance look requests are scheduled according to the order of the look requests in the queue. For look λ_i, the start time is $15(i-1)$ and the end time is $15i$. Surveillance looks occupy 100% of the window.

5.2 Example 2: Tracking looks in Condition I

In this example, the scheduler receives three tracking look requests, where the look parameters are shown in Table 5. It is evident that $t_1^* + l_1 < t_2^*$ and that $t_2^* + l_2 < t_3^*$. Therefore, the tracking looks are in Condition I and can be scheduled at their desired start times, as shown in the last column of Table 5.

Table 5: *Look parameters and start times for Example 2.*

	Look Parameters							*Start Time*
Look	s_n	t_n^*	u_n	l_n	B_n^*	δ_n	Δ_n	$\widehat{t_n}$
L_1	5	32	60	15	500	2	2	32
L_2	68	95	115	20	1000	1	1	95
L_3	197	220	240	15	200	1	1	220

Given a track schedule, the Gap-Filling Sub-Scheduler then schedules as many surveillance looks as possible in the idle time intervals. The resulting schedule is shown in Table 6. The column labeled 'function' indicates whether a time interval is idle or is occupied by a surveillance look or a tracking look. Of the 20 surveillance looks in the queue, 17 are scheduled during the window, so that surveillance look λ_{18} is at the top of the queue at the end of the window. Of the window length of 310 ms, 82.3% is occupied by surveillance looks, 16.1% by tracking looks, and 1.6% is idle.

For this example, the scheduling of the tracking looks is not affected by many of the look parameters, such as the scheduling interval $[s_n, u_n]$, the peak benefit B_n^* and the slopes for

early and late scheduling, δ_n and Δ_n. Examples 3 and 4 present cases where not all tracking looks can be scheduled at their desired start times. In these cases, the look parameters will determine which look requests are down-selected and when they are scheduled.

Table 6: *Output of Sequential Scheduler for Example 2.*

Function	Look	Start time (ms)	End time (ms)
Surveillance	λ_1	0	15
Surveillance	λ_2	15	30
Idle	-	30	32
Tracking	L_1	32	47
Surveillance	λ_3	47	62
Surveillance	λ_4	62	77
Surveillance	λ_5	77	92
Idle	-	92	95
Tracking	L_2	95	115
Surveillance	λ_6	115	130
Surveillance	λ_7	130	145
Surveillance	λ_8	145	160
Surveillance	λ_9	160	175
Surveillance	λ_{10}	175	190
Surveillance	λ_{11}	190	205
Surveillance	λ_{12}	205	120
Tracking	L_3	220	235
Surveillance	λ_{13}	235	250
Surveillance	λ_{14}	250	265
Surveillance	λ_{15}	265	280
Surveillance	λ_{16}	280	295
Surveillance	λ_{17}	295	310

5.3 Examples 3A and 3B: tracking looks in Condition II

Examples 3A and 3B consider a set of tracking looks that are in Condition II. The two examples have look requests which differ in their slopes for early and late scheduling to highlight the characteristics of the TSBF Sub-Scheduler.

5.3.1 Example 3A

In this example, the scheduler receives ten tracking look requests, where the look parameters are shown in Table 7. Recall the definitions of radar loading and look request con-

ditions from Section 3.2. For this set of looks, $\bar{l} = 195$ ms and $\tau = 255$ ms. Furthermore, it is not the case that $t_n^* + l_n < t_{n+1}^*$ for $n = 1, ..., 9$. Therefore, the tracking looks are in Condition II.

Table 7: *Look parameters, look metrics and start times for Example 3A.*

	Look Parameters							*Look Metrics*		*Start Time*
Look	s_n	t_n^*	u_n	l_n	B_n^*	δ_n	Δ_n	t_n'	E_n	$\widehat{t_n}$
L_1	5	25	40	15	700	5	5	5	35	18
L_2	15	38	56	15	200	1	1	20	36	33
L_3	21	48	68	20	1000	10	10	40	33	48
L_4	45	50	70	15	500	4	4	60	15	68
L_5	75	88	114	25	900	8	8	75	39	88
L_6	130	142	168	20	300	4	4	130	38	135
L_7	135	155	192	25	700	5	5	150	42	155
L_8	150	170	225	15	600	6	6	175	50	180
L_9	172	210	220	20	900	10	10	190	30	210
L_{10}	195	225	240	20	200	2	2	210	30	230

The TSBF Sub-Scheduler begins by carrying out the metrics calculation for the set of looks requests. The look metrics $\{t_n'\}$ and $\{E_n\}$ are shown in Table 7. As detailed in Section 3.4, $\{E_n\}$ specify the maximum delay that can be applied to each look within a sequence. Because the parameters $\{E_n\}$ are non-negative for all n, the set of look requests is viable. Recall that the Metrics Calculation Algorithm partitions the set of look requests into a number of sequences. In this case, the set of look requests was partitioned into two sequences; that is, $Q = 2$. The parameters D_q and G_q corresponding to the sequences are shown in Table 8. As specified in Section 3.4, G_q quantify the maximum delay that can be applied to looks which are in different sequences. It is seen that the first sequence includes the look requests L_1 through L_5, while the second sequence includes look requests L_6 through L_{10}. The parameter G_2 is undefined since the Metrics Calculation Algorithm always leaves G_Q undefined.

Table 8: *Sequence parameters for Example 3A.*

Sequence (q)	D_q	G_q
1	1	30
2	6	-

The set of look requests is viable, so look down-selection is not required and the peak benefit values have no effect on the final schedule produced by the TSBF Sub-Scheduler. The

next step for the TSBF Sub-Scheduler is the Start Time Assignment Algorithm. The start times $\widehat{t_n}$ computed by the TSBF Sub-Scheduler are shown in the last column of Table 7. The Start Time Assignment Algorithm uses the metrics $\{t'_n\}$ as initial start times and increments the start times to maximize the total benefit. Figure 5 shows the total benefit as the start times are incremented by the simplex algorithm. In this example, twelve iterations of the simplex method are required to compute the start times that maximize total benefit. As explained in Annex A, if the entering basic variable has zero value, then an iteration of the simplex method will result in no change in the objective function. This accounts for the iterations in Figure 5 where the objective function remains unchanged.

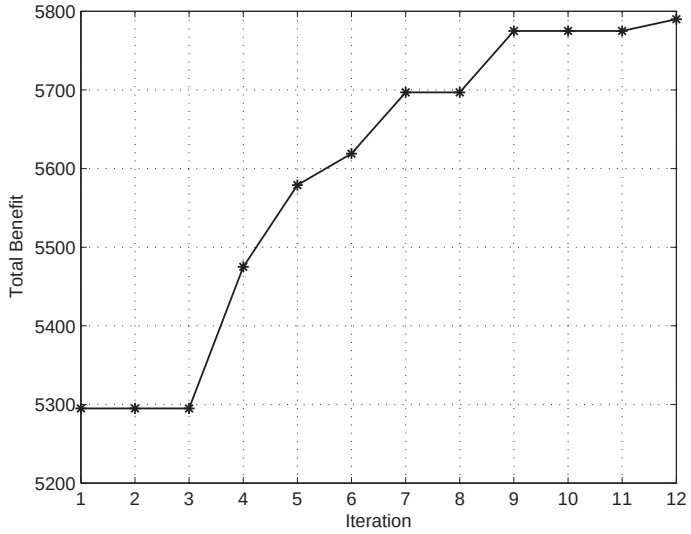

Figure 5: *Increase in total benefit for the simplex method for Example 3A.*

In this example, the two sequences of looks can be examined individually to understand how the total benefit is maximized. Consider the first sequence of looks, which consists of L_1 through L_5. For looks L_1, L_2, L_3 and L_5, $t'_n < t^*_n$ so that increasing the start time from t'_n increases the benefit function of these looks. However, for look L_4, $t'_n > t^*_n$ so that increasing the start time from t'_n decreases the benefit function of the fourth look. For each one millisecond increase to the start times, the net positive benefit is 20. Increasing the start times incrementally results in a net increase in total benefit until either one of the look start times reaches either its desired start time or its latest start time. In this case, this occurs when the start time for L_3 reaches 48 ms, which is its desired start time. Further increasing of the start times of the first four looks does not increase the total benefit. However, further increasing of the start time of the fifth look does increase the total benefit. The total benefit of the first sequence is maximized when the start time of the fifth look is the desired start time of 88 ms.

The second series of looks consists of looks L_6 through L_{10}. For each look except L_8,

$t'_n < t^*_n$ so that increasing the start time from t'_n increases the individual benefit function. For look L_8, $t'_n > t^*_n$ so that increasing the start time from t'_n decreases the benefit function of this look. Increasing the start times for the second sequence increases the total benefit until the start time for L_7 reaches 155 ms, which is its desired start time. At this point, there is no net increase in benefit from increasing the start times for L_6, L_7 and L_8. However, increasing the start times for L_9 and L_{10} increases the net benefit until the start time for L_9 reaches 210 ms, which is its desired start time.

Because the radar is underloaded with tracking look requests, the sub-scheduler has flexibility in assigning start times for Example 3A. It is seen that the values of the slopes for early and late scheduling, δ_n and Δ_n, determine the start times which maximize the total benefit. The two looks with the largest slope values, L_3 and L_9, are both scheduled at their desired start times. Note that having larger slope values does not necessarily mean that a given look will be scheduled closer to its desired start time than a neighboring look with smaller slope values. In this example, look L_7 is scheduled at its desired start time while look L_8 is not, even though $\delta_7 = \Delta_7 = 5$ while $\delta_8 = \Delta_8 = 6$.

Table 9: *Output of Sequential Scheduler for Example 3A.*

Function	Look	Start time (ms)	End time (ms)
Surveillance	λ_1	0	15
Idle	-	15	18
Tracking	L_1	18	33
Tracking	L_2	33	48
Tracking	L_3	48	68
Tracking	L_4	68	83
Idle	-	83	88
Tracking	L_5	88	113
Surveillance	λ_2	113	128
Idle	-	128	135
Tracking	L_6	135	155
Tracking	L_7	155	180
Tracking	L_8	180	195
Surveillance	λ_3	195	210
Tracking	L_9	210	230
Tracking	L_{10}	230	250
Surveillance	λ_4	250	265
Surveillance	λ_5	265	280
Surveillance	λ_6	280	295
Surveillance	λ_7	295	310

Given the look start times shown in the final column of Table 7, the Gap-Filling Sub-Scheduler fills the idle time intervals with surveillance looks. Table 9 shows the output of the Sequential Scheduler for Example 3A. Seven surveillance looks are scheduled. Of the total window, 61.3% is occupied by tracking looks, 33.9% by surveillance looks, and 4.8% is idle.

The idle time intervals produced by the Sequential Scheduler are less than the length of the surveillance looks. Therefore, shorter surveillance look lengths result in shorter idle time intervals. The total amount of idle time produced by the schedule depends on both the length of the surveillance looks and on the total number of idle intervals generated by the TSBF Sub-Scheduler.

5.3.2 Example 3B

In Example 3B, the scheduler receives ten tracking look requests. The look parameters are identical to those from Example 3A, except that δ_2, Δ_2, δ_8, and Δ_8 are increased to 12. These modified values are indicated in bold italics in Table 10. The change in these parameters results in modified start times for seven of the ten looks. The new start times are indicated in bold italics in the last column of Table 10.

Table 10: *Look parameters, look metrics and start times for Example 3B. Values in bold italics are variants from Example 3A.*

	Look Parameters							*Look Metrics*		*Start Time*
Look	s_n	t_n^*	u_n	l_n	B_n^*	δ_n	Δ_n	t_n'	E_n	$\widehat{t_n}$
L_1	5	25	40	15	700	5	5	5	35	***20***
L_2	15	38	56	15	200	***12***	***12***	20	36	***35***
L_3	21	48	68	20	1000	10	10	40	33	***50***
L_4	45	50	70	15	500	4	4	60	15	***70***
L_5	75	88	114	25	900	8	8	75	39	88
L_6	130	142	168	20	300	4	4	130	38	***130***
L_7	135	155	192	25	700	5	5	150	42	***150***
L_8	150	170	225	15	600	***12***	***12***	175	50	***175***
L_9	172	210	220	20	900	10	10	190	30	210
L_{10}	195	225	240	20	200	2	2	210	30	230

Due the increase in δ_2 and Δ_2, look L_2 is scheduled at 35 ms, which is 2 ms closer to its desired start time than in Example 3A. The neighboring looks L_3 and L_4 prevent L_2 from being scheduled at its desired start time of 33 ms. Look L_4 is scheduled at its latest start time of 70 ms, which forces L_3 to be scheduled to later than 50 ms and in turn forces L_2 to be scheduled no later than 35 ms.

Similarly, the increase in δ_8 and Δ_8 results in L_8 being scheduled closer to its desired start time. The neighboring looks L_6 and L_7 prevent L_8 from being scheduled at its desired start time of 170 ms. Looks L_6, L_7 and L_8 are scheduled at times t'_6, t'_7 and t'_8.

Examples 3A and 3B illustrate the effect of look parameters δ_n and Δ_n on the schedule computed by the TSBF Sub-Scheduler. Larger values of δ_n and Δ_n result in look L_n being scheduled closer to its desired start time.

5.4 Examples 4A and 4B: tracking looks in Condition III

Examples 4A and 4B consider a set of tracking looks that are in Condition III. The two examples have look requests with different values of peak benefit to highlight the characteristics of the TSBF Sub-Scheduler.

5.4.1 Example 4A

In this example, the scheduler receives twenty look requests, with look parameters as shown in Table 11. For this set of looks, $\bar{l} = 380$ ms and $\tau = 314$ ms, so that the set of looks is in Condition III. Therefore one or more look requests will need to be dropped to produce a viable set of looks.

The TSBF Sub-Scheduler begins by computing the look metrics for the set of look requests. These metrics which are calculated before down-selection are indicated in Table 11. The look requests are partitioned into one sequence; that is, $Q = 1$. It is seen that $E_n < 0$ for several looks. Therefore, the original set of look requests is not viable, and the sub-scheduler executes the Look Down-Selection Algorithm. As a result of the down-selection, looks L_7, L_{14}, L_{16} and L_{18} are dropped to produce a viable set of looks. The look metrics for the viable set after down-selection are shown in Table 11. After down-selection, $E_{10} = 0$, which indicates that for look L_{10}, $t'_{10} = u_{10}$. Look must be scheduled at time $t'_{10} = 165$ ms. All down-selected looks are scheduled at the times $\{t'_n\}$ which were calculated after down-selection. Therefore, the slopes for early and late scheduling, δ_n and Δ_n, have no effect on the final schedule that is generated.

In the original set of look requests, there were two look requests, L_{14} and L_{16}, with a peak benefit of 100, which is the lowest value of peak benefit among all look requests. Both of these look requests were dropped in look down-selection. Of the four look requests with a peak benefit of 200, two look requests were dropped. Therefore, the four look requests which were dropped had the lowest peak benefit values. This will not necessarily be the case for all sets of look requests, as look down-selection process is also affected the length of the scheduling intervals and the relationships between scheduling intervals among the different look requests.

Table 11: *Look parameters, look metrics and start times for Example 4A. The symbol 'X' indicates a look request that was dropped during down-selection.*

	Look Parameters							*Look Metrics before Down-Selection*		*Look Metrics after Down-Selection*		*Start Time*
Look	s_n	t_n^*	u_n	l_n	B_n^*	δ_n	Δ_n	t_n'	E_n	t_n'	E_n	$\widehat{t}_n$
L_1	0	19	46	20	300	2	2	0	46	0	46	0
L_2	3	29	61	20	200	6	6	20	41	20	41	20
L_3	10	36	62	15	500	9	9	40	22	40	22	40
L_4	20	45	74	25	300	9	9	55	19	55	19	55
L_5	46	77	107	25	400	3	3	80	27	80	27	80
L_6	93	119	153	15	500	5	5	105	48	105	48	105
L_7	90	120	146	20	200	4	4	120	26	X	X	X
L_8	106	131	164	25	500	6	6	140	24	120	44	120
L_9	104	138	173	20	700	3	3	165	8	145	28	145
L_{10}	111	139	165	20	300	2	2	185	-20	165	0	165
L_{11}	144	175	208	15	800	5	5	205	3	185	23	185
L_{12}	164	196	226	15	200	3	3	220	6	200	26	200
L_{13}	168	202	233	20	500	1	1	235	-2	215	18	215
L_{14}	172	206	238	20	100	10	10	255	-17	X	X	X
L_{15}	180	213	243	20	300	5	5	275	-32	235	8	235
L_{16}	192	224	253	15	100	10	10	295	-42	X	X	X
L_{17}	196	227	261	20	300	8	8	310	-49	255	6	255
L_{18}	214	240	269	15	200	1	1	330	-61	X	X	X
L_{19}	219	248	281	20	300	7	7	345	-64	275	6	275
L_{20}	241	270	299	15	300	7	7	365	-66	295	4	295

The schedule produced by the TSBF Sub-Scheduler has no idle intervals. Therefore, the tracking schedule is the final schedule, and no surveillance looks are scheduled in this example.

5.4.2 Example 4B

In this example, the scheduler receives twenty look requests. The look parameters are the same as in Example 4A, with the exceptions that the peak benefit for look L_9 is 400, and the peak benefits for looks L_4 and L_{14} are 1000. These modified values for peak benefit are indicated in bold italics in Table 12, which also presents the look metrics before and after down-selection and the start times $\widehat{t_n}$ generated by the TSBF Sub-Scheduler.

The last column in Table 12 shows the start time for the look requests, with the symbol 'X' indicating that the look was dropped. The start time entry is shown in bold italics if the outcome of the Look Down-Selection Algorithm was different from that in Example 4A. Looks L_2, L_{10} and L_{12} were down-selected in Example 4A but dropped in Example 4B. Looks L_7 and L_{14} were dropped in Example 4A but down-selected in Example 4B. Increasing the peak benefit of look L_{14} changed its outcome in the Look Down-Selection Algorithm. For all the other looks whose outcome of the Look Down-Selection Algorithm changed, their peak benefits were relatively low and unchanged from Example 4A. The changed peak benefit values of the other look requests caused the change in outcome of the Look Down-Selection Algorithm.

5.5 Summary

The examples presented in this section illustrate the schedules produced by the Sequential Scheduler for varying numbers of tracking look requests. When there are no tracking look requests or when all tracking look requests can be scheduled at their desired times, generating the tracking look schedule is trivial, as in Examples 1 and 2. However, when not all tracking look requests can be scheduled at their desired times, the TSBF Sub-Scheduler down-selects a viable set of looks if required, and schedules the viable set to maximize total benefit. Examples 3A and 3B showed that choosing larger values of δ_n and Δ_n for look L_n resulted in L_n being scheduled closer to its desired start time. Examples 4A and 4B showed that looks with larger values of peak benefit are more likely to survive look down-selection.

6 Conclusions

A technique for the scheduling of prioritized tracking and surveillance looks has been proposed. The Sequential Scheduler consists of the Two-Slope Benefit Function Sub-Scheduler for scheduling tracking looks and the Gap-Filling Sub-Scheduler for scheduling

Table 12: *Look parameters, look metrics and start times for Example 4B. The symbol 'X' indicates a look request that was dropped during down-selection. Peak benefit values which differ from Example 4A are shown in bold italics. Start time values are in bold italics if the outcome of look down-selection differed from that in Example 4A.*

	Look Parameters							*Look Metrics before Down-Selection*		*Look Metrics after Down-Selection*		*Start Time*
Look	s_n	t_n^*	u_n	l_n	B_n^*	δ_n	Δ_n	t_n'	E_n	t_n'	E_n	$\widehat{t_n}$
L_1	0	19	46	20	300	2	2	0	46	0	46	10
L_2	3	29	61	20	200	6	6	20	41	X	X	***X***
L_3	10	36	62	15	500	9	9	40	22	20	42	30
L_4	20	45	74	25	***1000***	9	9	55	19	35	39	45
L_5	46	77	107	25	400	3	3	80	27	60	47	70
L_6	93	119	153	15	500	5	5	105	48	93	60	95
L_7	90	120	146	20	200	4	4	120	26	108	38	***110***
L_8	106	131	164	25	500	6	6	140	24	128	36	130
L_9	104	138	173	20	***400***	3	3	165	8	153	20	155
L_{10}	111	139	165	20	300	2	2	185	-20	X	X	***X***
L_{11}	144	175	208	15	800	5	5	205	3	173	35	175
L_{12}	164	196	226	15	200	3	3	220	6	X	X	***X***
L_{13}	168	202	233	20	500	1	1	235	-2	188	45	190
L_{14}	172	206	238	20	***1000***	10	10	255	-17	208	30	***210***
L_{15}	180	213	243	20	300	5	5	275	-32	228	15	230
L_{16}	192	224	253	15	100	10	10	295	-42	X	X	X
L_{17}	196	227	261	20	300	8	8	310	-49	248	13	250
L_{18}	214	240	269	15	200	1	1	330	-61	X	X	X
L_{19}	219	248	281	20	300	7	7	345	-64	268	13	270
L_{20}	241	270	299	15	300	7	7	365	-66	288	11	290

surveillance looks. The TSBF Sub-Scheduler generates a schedule of tracking looks. The Gap-Filling Sub-Scheduler then fills in idle intervals in the resulting radar time line with surveillance looks.

To schedule tracking looks, the TSBF Sub-Scheduler computes metrics based on the look parameters associated with the set of input look requests. If required, a viable set of looks is down-selected from the original set of looks. The start times for the viable set are then chosen to maximize the total benefit of the set. For the two-slope benefit function, it is shown that the optimization is a linear program that can be solved using the simplex method. The processing time for one hundred look requests in on the order of one second, depending on the look parameters.

When the tracking look requests cannot be scheduled at their desired start times, the parameters of the benefit function determine whether looks are down-selected and when the looks are scheduled relative to their desired start times. If some looks must be dropped, larger values of peak benefit and larger scheduling intervals enhance the likelihood that a look will be down-selected. Larger values of slopes for early and late scheduling enhance the likelihood that a look will be scheduled closer to its desired start time. Examples were presented to illustrate these properties of the scheduler.

Previous work on task prioritisation can be used to specify the relative peak benefit values of the tracking look requests. However, further study is required to determine the optimal scheduling intervals and slopes for early and late scheduling.

References

[1] Stafford, W.K. (1990), Real-time control of a multifunction electronically scanned adaptive radar (MESAR), In *IEE Colloquium on on Real Time Management of Adaptive Radar Systems*, pp. 7/1–7/5.

[2] Vine, M.T. (1998), Fuzzy logic in radar resource management, In *IEE Colloquium on Multifunction Radar and Sonar Sensor Management Techniques*, pp. 1–4.

[3] Butler, J.M (1998), Multi-function radar tracking and control, Ph.D. thesis, UCL University of London.

[4] Sabatini, S. and Tarantino, M. (1994), Multifunction array radar - system design and analysis, Boston: Artech House.

[5] Strömberg, D. and Grahn, P. (1996), Scheduling of tasks in phased array radar, In *IEEE Int. Symp. Phased Array Systems and Radar*, Vol. 1, pp. 318–321.

[6] Izquierdo-Fuente, A. and Casar-Corredera, J.R. (1994), Optimal radar pulse scheduling using a neural network, In *IEEE Int. Conf. Neural Networks*, Vol. 7, pp. 4558–4591.

[7] Orman, A.J., C.N.Potts, Shahani, A.K., and Moore, A.R. (1996), Scheduling of tasks in phased array radar, *European Journal of Operational Research*, 90, 13–25.

[8] Duron, C. and Proth, J.-M. (2002), Multifunction radar: task scheduling, *Journal of Mathematical Modelling and Algorithms*, 1, 105–116.

[9] Miranda, S.L.C., C.J.Baker, Woodbridge, K., and Griffiths, H.D. (2007), Comparison of scheduling algorithms for multifunction radar, *IET Radar Sonar Navig.*, 1(6), 414–424.

[10] Dantzig, G.B. (1956), Recent advances in linear programming, *Management Science*, 2, 131–144.

[11] Fourer, R. (1985), A simplex algorithm for piecewise-linear programming I: derivation and proof, *Mathematical Programming*, 33, 204–233.

[12] Güder, F. and Nourie, F.J. (1996), A dual simplex algorithm for piecewise-linear programming, *Journal of the Operational Research Society*, 47, 583–590.

[13] Fourer, R. (1992), A simplex algorithm for piecewise-linear programming III: computational analysis and applications, *Mathematical Programming*, 53, 213–235.

[14] Miranda, S.L.C., C.J.Baker, Woodbridge, K., and Griffiths, H.D. (2007), Fuzzy logic approach for prioritization of radar tasks and sectors of surveillance in multifunction radar, *IET Radar Sonar Navig.*, 1(2), 131–141.

[15] Komorniczak, W., Kuczerski, T., and Pietrasinski, J. (2001), The priority assignment for detected targets in multifunction radar, *Journal of Telecommunications and InformationTechnology*, 1, 30–32.

[16] Noyes, S.P (1998), Calculation of next time for track update in the MESAR phased array radar, In *IEE Colloquium on Target Tracking and Data Fusion*, pp. 2/1–2/7.

[17] Davidson, G. (2007), Cooperation between tracking and radar resource management, In *IET International Conference on Radar Systems*, pp. 1–4.

[18] Berry, P.E. and Fogg, D.A.B. (2003), On the use of entropy for optimal radar resource management and control, In *2003 Proceedings of the International Radar Conference*, pp. 572–577.

[19] Pierre, D.A. (1986), Optimization theory with applications, New York: Dover.

Annex A: Simplex method

This annex shows how the simplex method for linear programming is applied to the optimisation problem presented in Section 3.6. The linear program is as follows. Choose $\{\alpha_n\}_1^N$ to maximize the objective function

$$\sum_{n=1}^{N} f_n(\alpha_n), \tag{A.1}$$

where $f_n(\alpha_n)$ is given by (11), subject to the constraints given in (12)-(14) and with the restrictions $\alpha_n, \phi_n, v_n, w_n, x_n \geq 0$. A feasible solution is a set of non-negative variables $\alpha_n, \phi_n, v_n, w_n, x_n$ that satisfy (12)-(14). The simplex method begins with an initial feasible solution and iteratively selects feasible solutions that increase the value of the objective function.

In this case, an initial feasible solution is given by $\alpha_n = 0$, for all n and $\phi_n = 0$ for all $n \in N_E$. Since $v_n = E_n - \alpha_n \geq 0$ for all n and $x_n = \phi_n - \alpha_n + t_n^* - t_n' \geq 0$ for all $n \in N_E$, it is evident that the variables α_n and ϕ_n are bounded, which shows that (A.1) is bounded and that a maximum exists.

To carry out the simplex method, variables are grouped into a set of nonbasic variables, which are set to zero, and a set of basic variables. For the initial feasible solution, the variables $\{\alpha_n\}_{n=1}^N$ and $\{\phi_n\}_{n\in N_E}$ are the nonbasic variables and the variables $\{v_n\}_{n=1}^N$, $\{w_n\}_{n=1}^N$ and $\{x_n\}_{n\in N_E}$ are the basic variables. The simplex method then iterates on the following two steps.

1. Determine the entering basic variable by computing which nonbasic variable, if allowed to take on a positive value, will increase the value of the objective function most rapidly.

2. Increase the value of the entering basic variable until one of the basic variables is forced to a value of zero. This basic variable is called the exiting basic variable.

The entering basic variable becomes a basic variable and the exiting basic variable becomes a nonbasic variable. Steps 1 and 2 are then repeated. The procedure stops when no entering basic variables exist, and the resulting values of the variables maximize the objective function.

If more than one variable qualifies as the entering basic variable, then the entering basic variable is chosen arbitrarily from amoung these candidates. If a basic variable has zero value, it may be necessary to exchange to choose this variable as the entering basic variable in order to proceed with the iteration. Such an operation will result in the objective function remaining constant for an iteration. This is illustrated in Figure 5.

This page intentionally left blank.

Annex B: Other forms of the benefit function

For the TSBF Sub-Scheduler, the benefit function was defined as a two-slope function, as given by (1) and (2). This appendix examines other forms of the benefit function which may be considered.

The benefit function can be specified as an order-r function, which is defined by

$$B_n(t_n) = B_n^* - c_n \|t_n - t_n^*\|^r, \tag{B.1}$$

where $c_n \in \Re$, $c_n \geq 0$, and $r \geq 2$ is an integer. Order-r benefit functions with $r = 2$, $r = 3$, and $r = 4$ are shown in Figure B.1, when $c_n = 1$. It is seen that the benefit function decreases more slowly in $\|t_n - t_n^*\|$ as r increases. Given a viable set of N looks with order-r benefit

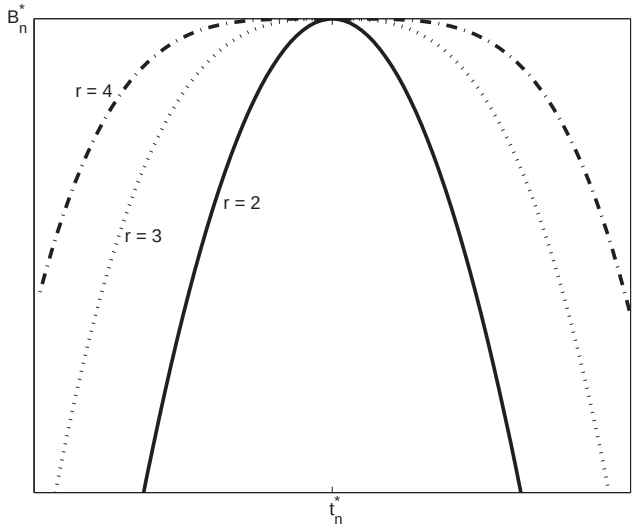

Figure B.1: *Order-r benefit functions for* $c_n = 1$.

functions, the Start Time Assignment Algorithm seeks to choose $\{\alpha_n\}_1^N$ to minimize

$$\sum_{n=1}^{N} c_n \|\alpha_n - (t_n^* - t_n')\|^r, \tag{B.2}$$

subject to the constraints (4) and (5). This optimisation can be carried out using Lagrange multipliers and solving for the Kuhn-Tucker conditions for optimality [19]. Solving this non-linear program requires significantly more computation complexity than that of the simplex method.

Piecewise-linear benefit functions with multiple slopes may also be considered. These functions are more general than two-slope functions, while the process for selecting start times is still a linear program. However, in order to apply the simplex method, additional variables must be introduced to the problem. In general, one additional variable will need to be introduced for each additional linear piece of each benefit function [11].

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

   Defence R&D Canada – Ottawa
   3701 Carling Avenue, Ottawa, Ontario, Canada
   K1A 0Z4

2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)

   UNCLASSIFIED

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

   Scheduling for multifunction radar via two-slope benefit functions

4. AUTHORS (Last name, followed by initials – ranks, titles, etc. not to be used.)

   Moo, P.W.

5. DATE OF PUBLICATION (Month and year of publication of document.)

   December 2010

6a. NO. OF PAGES (Total containing information. Include Annexes, Appendices, etc.)

   56

6b. NO. OF REFS (Total cited in document.)

   19

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

   Technical Memorandum

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

   Defence R&D Canada – Ottawa
   3701 Carling Avenue, Ottawa, Ontario, Canada K1A 0Z4

9a. PROJECT NO. (The applicable research and development project number under which the document was written. Please specify whether project or grant.)

   11as04

9b. GRANT OR CONTRACT NO. (If appropriate, the applicable number under which the document was written.)

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)

   DRDC Ottawa TM 2010-259

10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

   (X) Unlimited distribution
   ( ) Defence departments and defence contractors; further distribution only as approved
   ( ) Defence departments and Canadian defence contractors; further distribution only as approved
   ( ) Government departments and agencies; further distribution only as approved
   ( ) Defence departments; further distribution only as approved
   ( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11)) is possible, a wider announcement audience may be selected.)

   Full unlimited announcement.

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

The scheduling of tracking and surveillance looks for multifunction radar is considered. A sequential technique is proposed, whereby tracking looks are scheduled first, and surveillance looks are then scheduled to occupy gaps in the radar time line. The Two-Slope Benefit Function (TSBF) Scheduler is used to schedule the tracking looks and requires that each tracking look request has a benefit function, which specifies benefit as a function of start time. This method accounts for both look priority and target dynamics in formulating a look schedule. If the radar is overloaded with tracking look requests, the TSBF Scheduler down-selects a set of looks which can be scheduled, using a method which favours higher priority looks. Looks are scheduled to maximize the total benefit, and the resulting maximization is shown to be a linear program which can be solved efficiently using the Simplex Method. Given a tracking look schedule, the Gap-Filling Scheduler is used to schedule surveillance looks. A method for prioritising surveillance looks is proposed.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Radar resource management
Task scheduling
Simplex method

Defence R&D Canada

Canada's leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

www.drdc-rddc.gc.ca